Interactions between Neural ODE, Bayesian Inverse Problems and Optimal Transportation, applied to the Calderón problem

Pablo Angulo

Valladolid, NBMDS, November 11th, 2021

Outline of the talk

- Neural ODEs
- Continuous Normalizing Flows
- Differentiable Programming, Scientific Machine Learning, `julia`
- The bayesian approach to Inverse Problems
- Informative priors for the Calderón Problem
- Open questions and open problems

Outline of the talk

- Neural ODEs
- Continuous Normalizing Flows
- Differentiable Programming, Scientific Machine Learning, `julia`
- The bayesian approach to Inverse Problems
- Informative priors for the Calderón Problem
- Open questions and open problems

Outline of the talk

- Neural ODEs
- Continuous Normalizing Flows
- Differentiable Programming, Scientific Machine Learning, `julia`
- The bayesian approach to Inverse Problems
- Informative priors for the Calderón Problem
- Open questions and open problems

Outline of the talk

- Neural ODEs
- Continuous Normalizing Flows
- Differentiable Programming, Scientific Machine Learning, `julia`
- The bayesian approach to Inverse Problems
- Informative priors for the Calderón Problem
- Open questions and open problems

Outline of the talk

- Neural ODEs
- Continuous Normalizing Flows
- Differentiable Programming, Scientific Machine Learning, `julia`
- The bayesian approach to Inverse Problems
- Informative priors for the Calderón Problem
- Open questions and open problems

Outline of the talk

- Neural ODEs
- Continuous Normalizing Flows
- Differentiable Programming, Scientific Machine Learning, `julia`
- The bayesian approach to Inverse Problems
- Informative priors for the Calderón Problem
- Open questions and open problems

ODENets and Neural ODEs

Residual Networks (ResNets), and also recurrent neural network decoders, normalizing flows and other architectures follow this scheme:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + f(\mathbf{x}_t, \theta_t)$$

Neural ODE (also ODEnet) Replace those schemes by the solution of dynamical systems:

$$\dot{\mathbf{x}} = f(\mathbf{x}, t, \theta)$$

Weinan 17, Haber Ruthotto 17-18, Chen et al 18-19

Neural ODE are also a system of ODEs defined by a neural network:

$$\dot{\mathbf{x}} = NN(\mathbf{x}, t, \theta)$$

Rackauckas et al 20 Universal Differential Equations for Scientific Machine Learning

ODENets and Neural ODEs

Residual Networks (ResNets), and also recurrent neural network decoders, normalizing flows and other architectures follow this scheme:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + f(\mathbf{x}_t, \theta_t)$$

Neural ODE (also ODEnet) Replace those schemes by the solution of dynamical systems:

$$\dot{\mathbf{x}} = f(\mathbf{x}, t, \theta)$$

Weinan 17, Haber Ruthotto 17-18, Chen et al 18-19

Neural ODE are also a system of ODEs defined by a neural network:

$$\dot{\mathbf{x}} = NN(\mathbf{x}, t, \theta)$$

Rackauckas et al 20 Universal Differential Equations for Scientific Machine Learning

ODENets and Neural ODEs

Residual Networks (ResNets), and also recurrent neural network decoders, normalizing flows and other architectures follow this scheme:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + f(\mathbf{x}_t, \theta_t)$$

Neural ODE (also ODEnet) Replace those schemes by the solution of dynamical systems:

$$\dot{\mathbf{x}} = f(\mathbf{x}, t, \theta)$$

Weinan 17, Haber Ruthotto 17-18, Chen et al 18-19

Neural ODE are also a system of ODEs defined by a neural network:

$$\dot{\mathbf{x}} = NN(\mathbf{x}, t, \theta)$$

Rackauckas et al 20 Universal Differential Equations for Scientific Machine Learning

Software

- `torchdiffeq` for `python`
- `FluxDiffEq` for `julia`
- Embedded in a machine learning toolkit (`torch` / `Flux`)$\Rightarrow$ runs in parallel, CPU or GPU
- Automatic Differentiation
- Fast implementations of common operations and optimization algorithms
- $\Rightarrow$ Easy parameter estimation

Julia

`julia` is pretty awesome for numerical computation

- open source
- general purpose
- dynamic
- compiled
- macros as in LISP (`@. sin(xs)/2`)
- source-to-source Automatic Differentiation
- fast!

Composition

Two different ideas:

- A neural network that defines an ODE:

```
# Make a neural net with a NeuralODE layer
dudt = FastChain(
    (x, p) -> x.^3, # Guess a cubic function
    # Multilayer perceptron for the part we don't know
    FastDense(2, 50, tanh), FastDense(50, 2) )
prob_neuralode = NeuralODE(dudt, tspan, Tsit5(), u0, saveat = ts)
```

- Use an ODE layer as part of a neural network structure.

```
# MNIST: 784 pixels to 10 digits
down = Chain(Flux.flatten, Dense(784, 20, tanh)) |> gpu

nn = Chain(Dense(20, 10, tanh),
           Dense(10, 10, tanh),
           Dense(10, 20, tanh)) |> gpu

nn_ode = NeuralODE(nn, (0.0, 1.0), Tsit5(),
                   save_everystep = false,
                   reltol = 1e-3) |> gpu

fc  = Chain(Dense(20, 10)) |> gpu

# Build our overall model topology
model = Chain(down,
              nn_ode,
              DiffEqArray_to_Array,
              fc) |> gpu;
```

Julia ecosystem for Scientific Machine Learning

There is actually a healthy community centered on building and using `julia` tools for

Universal Differential Equations

Models composed of

- ODEs
- Delay Differential Equations
- Stochastic ODEs
- Partial Differential Equations
- Differential Algebraic Equations
- Events (such as “ball bounces in floor”)
- Neural Networks

where the free parameters minimize some regularized loss wrt data.

The idea is to put all our knowledge into the model, and “fill the gaps” with free parameters and neural networks

$$\begin{aligned} \dot{x} &= \alpha x + U_1(x, y) \\ \dot{y} &= -\delta y + U_2(x, y) \end{aligned}$$

Rackauckas et al 20 Universal Differential Equations for Scientific Machine Learning

Julia ecosystem for Scientific Machine Learning

There is actually a healthy community centered on building and using `julia` tools for

Universal Differential Equations

Models composed of

- ODEs
- Delay Differential Equations
- Stochastic ODEs
- Partial Differential Equations
- Differential Algebraic Equations
- Events (such as “ball bounces in floor”)
- Neural Networks

where the free parameters minimize some regularized loss wrt data.

The idea is to put all our knowledge into the model, and “fill the gaps” with free parameters and neural networks

$$\begin{aligned} \dot{x} &= \alpha x + U_1(x, y) \\ \dot{y} &= -\delta y + U_2(x, y) \end{aligned}$$

Rackauckas et al 20 Universal Differential Equations for Scientific Machine Learning

Parameter estimation of an ODE in julia

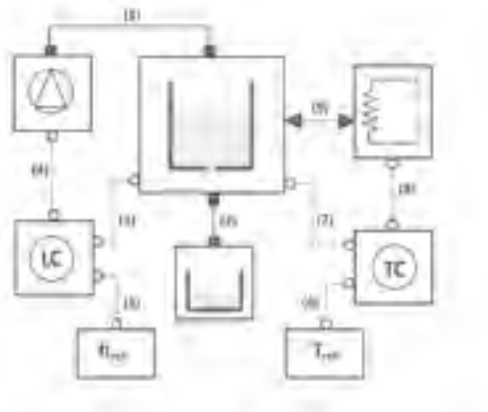
```
using DifferentialEquations, DiffEqFlux

function lotka_volterra(u, p, t)
    [ α, β, δ, γ ] = p
    [ x, y ] = u
    [ α*x - β*x*y, - δ*y + γ*x*y ]
end

prob = ODEProblem(lotka_volterra, u0, tspan, p)
sol = solve(prob, Tsit5())

function loss(p)
  sol = solve(prob, Tsit5(), p=p, saveat = tsteps)
  sum(abs2, sol - data)
end

result_ode = DiffEqFlux.sciml_train(loss, p)
```

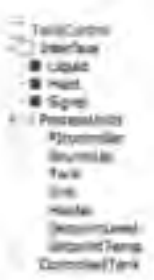
Finding interaction terms of an ODE in julia

```
U = FastChain(
    FastDense(2,5,rbf), FastDense(5,5, rbf),
    FastDense(5,5, rbf), FastDense(5,2)
)
# Get the initial parameters
p = initial_params(U)

# Define the hybrid model
function lotka_volterra_ude(du, u, p, t)
    Up = U(u, p) # Network prediction
    [ α*u[1] + Up[1], - β*u[2] + Up[2]]
end

# Define the problem
prob_nn = ODEProblem(lotka_volterra_ude, u0, tspan, p)
```

(aside) modelica

The idea os SciML tickles me, because many people in the Naval Industry in Spain is very excited with modeling tools like the language `modelica`:



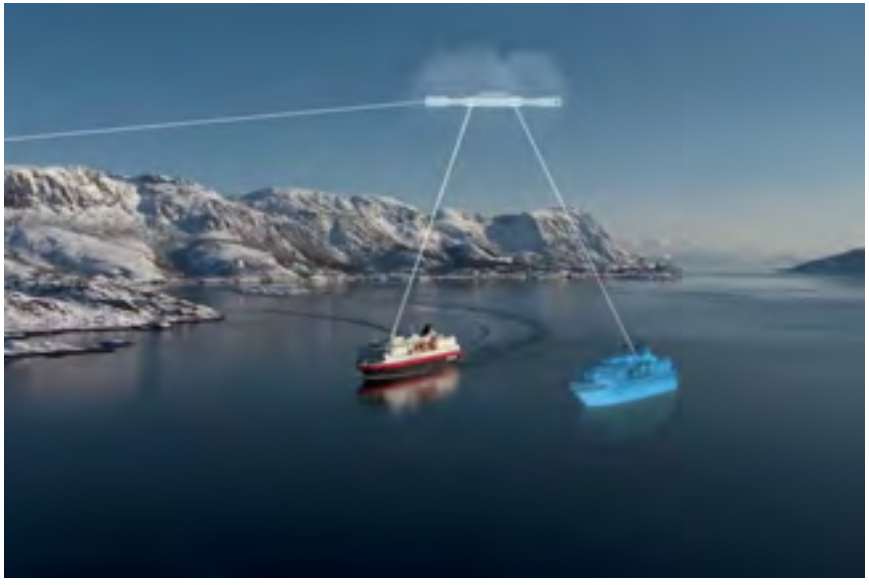


```
model Capacitor "Ideal capacitor"
  import Modelica.Units.SI;
  Pin p, n;
  parameter SI.Capacitance C "Capacitance";
protected
  SI.Current i "Current entering p, leaving n";
  SI.Voltage u "Voltage drop (= p.u - n.u)";
equation
  u = p.u - n.u;
  C*der(u) = i;
  i = p.i;
  i = -n.i;
end Capacitor;
```

The idea is precisely to define models for every subsystem of a ship. And each such system will need some parameter estimation, or model inference.

(aside) Digital twins in the Naval Industry

And the whole industry is even more crazy about **digital twins**.



mfame.guru

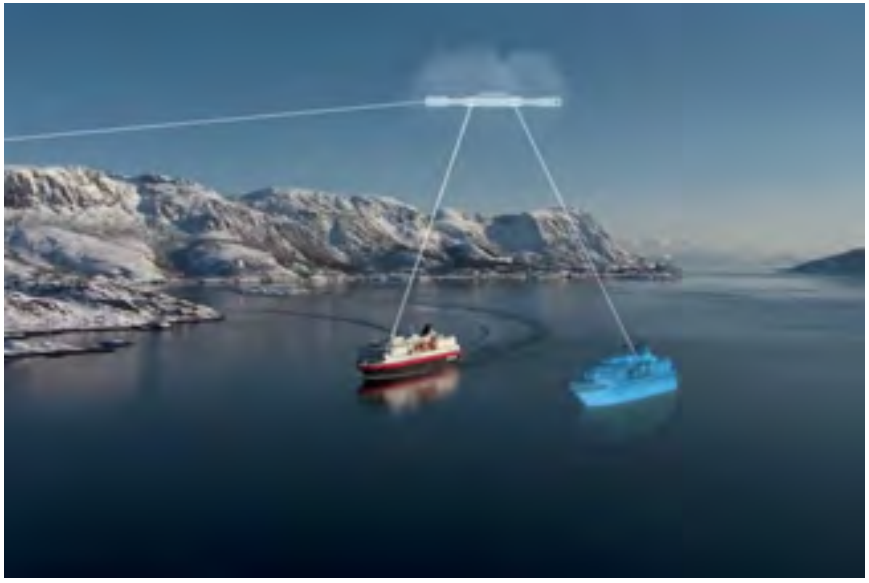
Digital Twin (a personal definition)

A folder full of models for different subsystems together with

- a presentation layer that transmits coherence, unity, purpose...
- sensors to collect data from real operations
- *the possibility of updating the model parameters with that data*

(aside) Digital twins in the Naval Industry

And the whole industry is even more crazy about **digital twins**.



mfame.guru

Digital Twin (a personal definition)

A folder full of models for different subsystems together with

- a presentation layer that transmits coherence, unity, purpose...
- sensors to collect data from real operations
- *the possibility of updating the model parameters with that data*

Regularization

A well-posed problem can be solved by minimizing the empirical error:

$$\min_{\theta} \sum \mathit{loss}\left(f(x_j, \theta), y_j\right)$$

In the presence of nonuniqueness or lack of stability, some amount of regularization is required:

$$\min_{\theta} \sum \mathit{loss}\left(f(x_j, \theta), y_j\right) + R(\theta)$$

If the objective is to recover some function, Gaussian processes are common: $R(\theta)$ is the norm of f_θ in the Reproducing Kernel Hilbert Space (RKHS) associated to the Gaussian process.

Regularization

A well-posed problem can be solved by minimizing the empirical error:

$$\min_{\theta} \sum \mathit{loss}\left(f(x_j, \theta), y_j\right)$$

In the presence of nonuniqueness or lack of stability, some amount of regularization is required:

$$\min_{\theta} \sum \mathit{loss}\left(f(x_j, \theta), y_j\right) + R(\theta)$$

If the objective is to recover some function, Gaussian processes are common: $R(\theta)$ is the norm of f_θ in the Reproducing Kernel Hilbert Space (RKHS) associated to the Gaussian process.

Bayesian Inverse Problems

Regularization admits a *"bayesian interpretation"*.

$$\min_\theta \underbrace{\sum_j \mathit{loss}\left(f(x_j,\theta),y_j\right)}_{\text{data likelihood}} \underbrace{+R(\theta)}_{\text{prior}}$$

The regularized loss to-be-minimized is the *formal?* negative logarithm of the posterior density.

$$\min_\theta \sum \mathit{loss}\left(f(x_j,\theta),y_j\right)+R(\theta) = \max_\theta \exp\left\{-\sum \mathit{loss}\left(f(x_j,\theta),y_j\right) - R(\theta)\right\}$$

In other words, the *Maximum a posteriori (MAP)* estimator for the density:

$$\text{MAP} = \text{argmax}_\theta \exp\left\{-\sum \mathit{loss}\left(f(x_j,\theta),y_j\right) - R(\theta)\right\}$$

Caveat minimizanti

But a function is not the same as a probability measure. A function is all you need to pose a minimization problem, but it is not a probability measure.
In **Bayesian Statistics**, *any unknown quantity is assigned a probability measure*. This is the rigorous way of measuring uncertainty...
... and this is the only “axiom” of Bayesian Statistics.
Priors are not plugged into the bayesian paradigm: they arise naturally as a consequence of Bayes Theorem.

Stuart 2010. Inverse problems: A Bayesian perspective

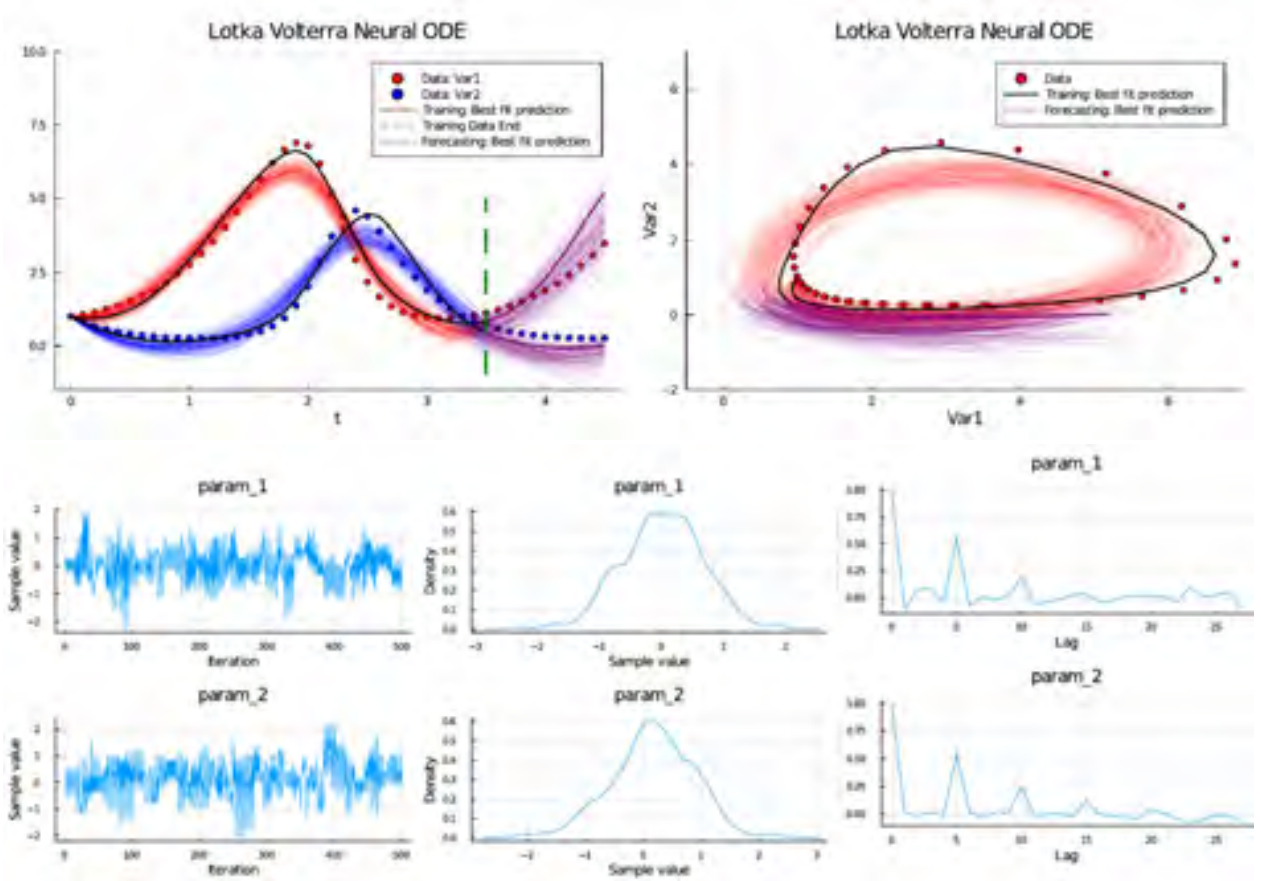
Caveat minimizanti

But a function is not the same as a probability measure. A function is all you need to pose a minimization problem, but it is not a probability measure.
In **Bayesian Statistics**, *any unknown quantity is assigned a probability measure*. This is the rigorous way of measuring uncertainty...
... and this is the only “axiom” of Bayesian Statistics.
Priors are not plugged into the bayesian paradigm: they arise naturally as a consequence of Bayes Theorem.

Stuart 2010. Inverse problems: A Bayesian perspective

Bayesian estimation of ODE parameters

And the `julia` SciML community has sharp tools for bayesian estimation of parameters of ODEs using NUTS and other MCMC variants:



Dandekar et al 21 Bayesian Neural Ordinary Differential Equations

"Bayesian Statistics using Julia and Turing"

(back to NeuralODE) Applications

- Drop-in replacements for ResNets
- Generative models
- Uncertainty quantification
- Time series with non uniformly spaced data
- Continuous Normalizing Flows for building Invertible Neural Networks
- Latent space interpolation
- Feature transfer

After the hype died out, it turns out that most of the things you can do with NeuralODE can be done with an alternative method.
But it remains an interesting tool, suitable for machine learning and amenable to mathematical analysis.

(back to NeuralODE) Applications

- Drop-in replacements for ResNets
- Generative models
- Uncertainty quantification
- Time series with non uniformly spaced data
- Continuous Normalizing Flows for building Invertible Neural Networks
- Latent space interpolation
- Feature transfer

After the hype died out, it turns out that most of the things you can do with NeuralODE can be done with an alternative method.
But it remains an interesting tool, suitable for machine learning and amenable to mathematical analysis.

Discrete Normalizing Flows

Goal: An invertible transformation carries a sample from an unknown distribution to a sample from a well known distribution. For instance, z_0 is normally distributed, z_n is the data.

$$z_1 = f_1(z_0), z_2 = f_2(z_1), ...$$

Example: **planar normalizing flow**

$$z_{t+1} = z_t + uh(w^T z(t) + b)$$

Problems were found, and have been adressed:

- need to compute jacobians
- use transformations that can be parameterized, but are always invertible
- stability issues

Discrete Normalizing Flows

Goal: An invertible transformation carries a sample from an unknown distribution to a sample from a well known distribution. For instance, z_0 is normally distributed, z_n is the data.

$$z_1 = f_1(z_0), z_2 = f_2(z_1), ...$$

Example: **planar normalizing flow**

$$z_{t+1} = z_t + uh(w^T z(t) + b)$$

Problems were found, and have been adressed:

- need to compute jacobians
- use transformations that can be parameterized, but are always invertible
- stability issues

Continuous Normalizing Flow

For a continuous-time transformation and an initial fixed density p_0, flow the z-space:

$$\dot{z} = f(z(t), t)$$

Then

$$\frac{\partial \log(p(z(\cdot, t)))}{\partial t} = \text{div}\left(\frac{\partial f}{\partial z}\right)$$

The push forward by the time-1 flow $p_1\lambda = Flow_0^1 \#(p_0\lambda)$ of an absolutely continuous probability measure $p_0\lambda$ (λ is the Lebesgue measure) is given by

$$p_1 = |Jac(Flow_0^1)|p_0 = exp\left(\int_0^1 \text{div}\, f(z, t)\right) p_0$$

Wasserstein distance of metrics

Wasserstein distance between two measures, à la Monge

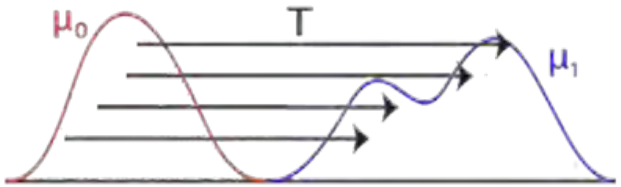
$$\mathcal{W}_2(\mu,\nu)^2 = \min_{T \text{ st } T\#\mu=\nu} \int_X \|x - T(x)\|^2 d\mu(x)$$

Kantorovich formulation

$$\mathcal{W}_2(\mu,\nu)^2 = \min_{\sigma\in\Gamma(\mu,\nu)} \int_{X\times X} \|x - y\|^2 d\sigma(x,y)$$

where $\Gamma(\mu,\nu) = \{\sigma \in \mathcal{M}(X\times X) : (\pi_{X\times\_})\#\sigma = \mu, (\pi_{\_\times X})\#\sigma = \nu)$

Intuition: Earth's mover distance



Benamou-Brenier formulation

Benamou-Brenier formulation of optimal transport

Equivalent formulation of optimal transport for two absolutely continuous measures with densities p_0 and p_T.

$$\mathcal{W}_2(p_0, p_1)^2 = \min_{f \in C(p_0, p_1)} \int \left(\int_0^1 \|f(z(t), t)\|^2 \, dt \right) p_0(z(0)) dz_0$$

$$C(p_0, p_1) = \left\{ Flow(f)_0^1 \# p_0 = p_1 \right\}$$

Solve this extended ode in just one pass, for a representative sample of initial points:

$$\begin{cases} \dot{z} = f(z, t) \\ \dot{j} = \mathrm{div}(f)(z, t) \\ \dot{r} = \|f(z, t)\|^2 \end{cases}$$

and compute both $\left(Flow_0^1(f) \# p_0\right)(z_0) = exp(j_{z_0}(1)) p_0(z_0)$ and

$$\int r_{z_0}(1) dt \; p_0(z_0) \; dz_0$$

Continuous Normalizing Flow as Optimal Transport

Minimize the loss regularized with a Benamou-Brenier cost to get a Wasserstein metric regularization term

$$\min_f \sum \text{datafit}\,(p_1, \text{data}) + \Lambda \mathcal{W}_2(p_0, p_1)^2$$

For instance $\text{datafit}\,(p_1\lambda, p_{\text{true}})$ could be a Kullback-Leibler divergence

$$\text{datafit} = KL(p_1\lambda, p_{\text{true}}) \approx const - \frac{1}{N}\sum \log(p_1(x_i))$$

Finlay et al 20 proposed to add also a "kinetic energy" regularization:

$$\begin{aligned} \min_f \quad & \sum \text{datafit}\,(p_1, \text{data}) + \Lambda_1 \mathcal{W}_2(p_0, p_1)^2 \\ & + \Lambda_2 \int \left(\int_0^1 \|\nabla f(z(t), t)\|^2 \, dt \right) p_0(z(0)) dz_0 \end{aligned}$$

Esteve et al 20 proposed $\|[b_t, w_t]\|_{H^k}$ as regularization, and studied the sensitivity to the final time.

Finlay et al 20, How to train your neural ODE the world of Jacobian and kinetic regularization

Esteve Geshkovski Pighin Zuazua 20 Large-time asymptotics in deep learning

Robustness

Neural ODE seems, empirically, to be more robust than CNNs, specially with $\mathcal{W}_1$ regularization.

Yan et al 20 On Robustness of Neural Ordinary Differential Equations

Some variants have been proved to be stable

- f is a gradient flow
- f is a Hamiltonian flow with dissipation

$$\begin{aligned} \dot{q} &= p \\ \dot{p} &= -\alpha q - \partial_q \varepsilon(q, x_0, \theta) \end{aligned}$$

Massaroli et al 20 Stable Neural Flows

Continuous Normalizing Flows for Density Estimation

Density estimation

- Given sample $\{x_i\}$.
- Goal: estimate the density

Flow from a reference distribution p_0 (usually $z \sim \mathcal{N}(0, 1)$) to the true x distribution p_1.
Loss is $-\sum \log(p_1(x_i))$, which admits interpretations both as the density of the sample, and the Kullback-Leibler divergence of the empirical distribution and p_1.

Generative Model Sample z at random and flow them $\Rightarrow$ get random x

Latent space interpolation Given x_1, x_2, usually $\alpha x_1 + (1 - \alpha)x_2$ is not meaningful, but $\alpha z_1 + (1 - \alpha)z_2$ often is.

INN for Inverse Problems

Invertible Neural Networks have been applied to inverse problems in a systematic way.
CNFs could be used in a totally equivalent way:

- Well-posed forward problem $x \to y$, can run it on arbitrary inputs $x \in \mathbb{R}^n$. Get $y \in \mathbb{R}^m$, with $n > m$.
- Goal: the inverse map $y \to x$

1. Draw x_i from a prior distribution.
2. Run the forward process to produce the dataset $\{(x_i, y_i)\}$.
3. Enlarge y with dummy variables $z_i \in \mathbb{R}^{n-m}$, random samples of a reference distribution, usually $z \sim \mathcal{N}(0, 1)$.
4. Learn

$$\text{Flow}(x) = (y, z)$$

Loss is a supervised loss ($\sum_i (y_i - F_y(x_i))^2$) plus a discrepancy between p_1 and the product of the empirical y distribution and a gaussian distribution for z.

INN for Inverse Problems

- Goal: the inverse map $y \to x$

$$\text{Flow}(x) = (y, z)$$

The map $x \to y$ is well posed, and now $(z, y) \to x$ is well posed too.

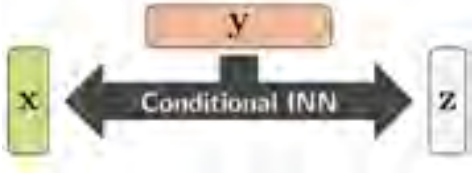
Feature transfer Given y_1, x_2, compute $(y_2, z_2) = \mathit{Flow}(x_2)$, and output $\mathit{Flow}^{-1}(y_1, z_2)$

Uncertainty Quantification Given y, sample z_i and get a sample $\{\mathit{Flow}^{-1}(z_i, y)\}$ for x.

Ardizzone et al 19 Guided Image Generation with Conditional Invertible Neural Networks

Conditional INN

Alternative: conditional invertible neural networks



This has not yet been carried to Neural ODE, but it could be done.
Loss function is minus log of the posterior probability of the sample.

Generation of alternatives

Feature transfer

Ardizzone et al 19 Guided Image Generation with Conditional Invertible Neural Networks

Calderón problem

In an open set Ω of $\mathbb{R}^2$ or $\mathbb{R}^3$, given conductivities γ, the voltages satisfy the equation:

$$-\nabla \cdot (\gamma \nabla u) = 0$$

And a well posed *forward problem* is to predict the currents at $\partial\Omega$ given the voltages at $\partial\Omega$.
The inverse problem is more interesting: given measurements from experiments where the voltage is set on the boundary and the currents are measured, recover the conductivities in Ω.

Definition (Dirichlet to Neumann map)

$\Lambda : H^{\frac{1}{2}}(\partial\Omega) \to H^{-\frac{1}{2}}(\partial\Omega)$ defined by

$$\Lambda(f) = \frac{\partial u}{\partial n}|_{\Omega}$$

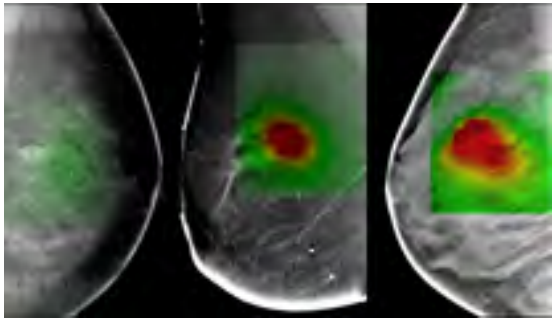
where u is the unique solution to the **boundary value problem**

$$\begin{cases} -\nabla \cdot (\gamma \nabla u) = 0 & \Omega \\ u = f & \partial\Omega \end{cases}$$

Applications

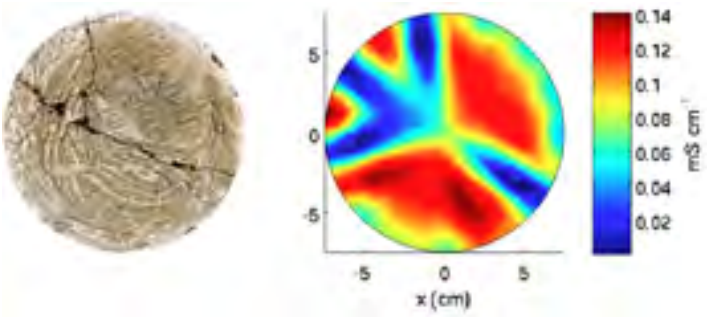
Initial motivation: finding *underground petrol* deposits through *electrical imaging* (and *seismic imaging* poses a similar problem).

Cancer detection



Kim et al 07

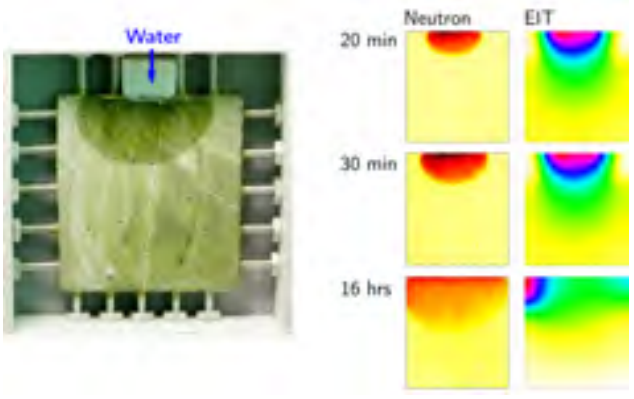
Cracks in structures



Karhunen et al 10

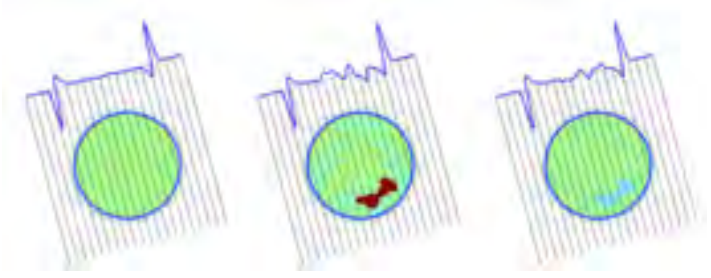
Also

Water in concrete structures



Hallaji et al 15

Stroke detection



Agnelli et al 21

Numerical methods based on CGO solutions

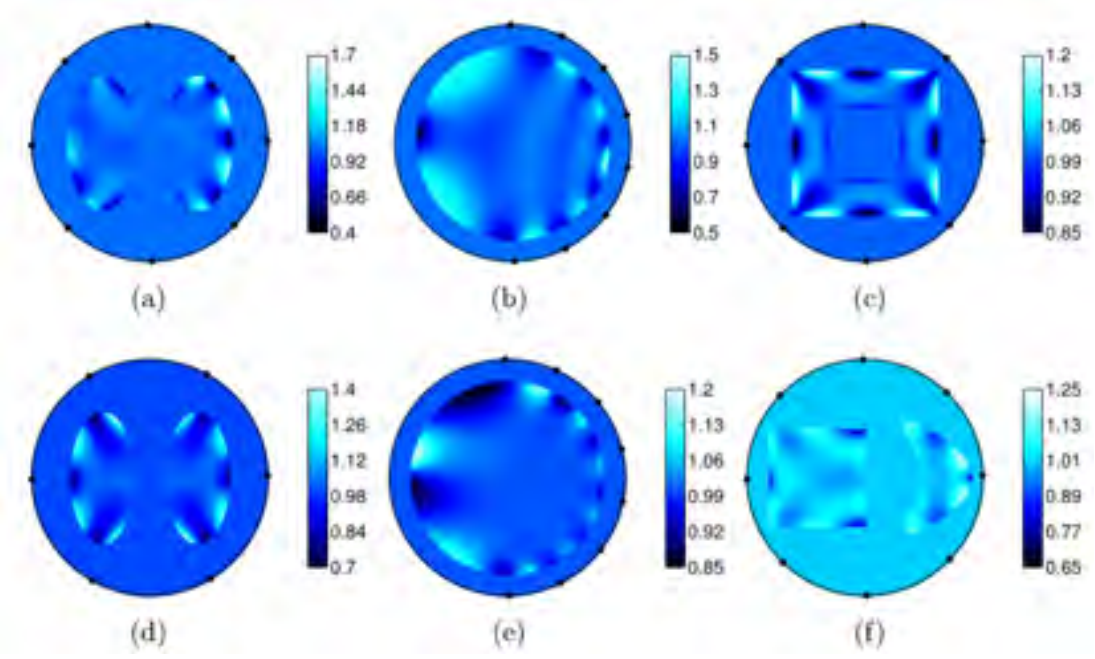
- 81 Calderón rediscovered the problem.
- 87 $n = 3$ Sylvester-Uhlmann: $\rho \in C^2$ Introduced *CGO* solutions $u = e^{\varphi(x)+i\psi}(1 + r)$
- 96 Nachman: recover smooth conductivities
- 00 Siltanen, Mueller and Isaacson: Numerical CGOs
- 06 Astala and Päivärinta: Beltrami-type solutions recover a bounded potential
- 10 Astala, Mueller, Päivärinta and Siltanen: First numerical solution method
- 11 Astala, Mueller, Päivärinta, Perämäki and Siltanen: New EIT reconstruction method
- 14 Astala, Päivärinta, Reyes and Siltanen: Numerical experiments with discontinuous conductivities

Ill-posed: ghosts

But the Calderón problem is notoriously *ill-posed*: quite different conductivities give rise to similar Dirichlet to Neumann data.
Low-pass filters mitigate, but do not eliminate this problem.
In reality, we don't know the full Λ, but we can only set the voltages and measure the currents at a few electrodes.
The result are **ghosts**: nontrivial conductivities completely undetectable by electrical tomography.
And it's also computationally challenging.

Ill-posed: ghosts

But the Calderón problem is notoriously *ill-posed*: quite different conductivities give rise to similar Dirichlet to Neumann data.
Low-pass filters mitigate, but do not eliminate this problem.
In reality, we don't know the full Λ, but we can only set the voltages and measure the currents at a few electrodes.
The result are **ghosts**: nontrivial conductivities completely undetectable by electrical tomography.
And it's also computationally challenging.



Chesnel et al 14

Priors for Inverse Problems

Gaussian processes are the favorite priors for linear inverse problems.

Stuart 2010. Inverse problems: A Bayesian perspective

Usually, only the maximum a posteriori (MAP) estimator is kept, so the problem is just minimization with a regularization term.

For some authors, the “bayesian approach” to inverse problems is just minimization of a regularized loss.

Some authors also prove frequentist properties of this MAP estimator.

Monard Nickl Paternain 2017 Efficient Nonparametric Bayesian Inference For X-Ray Transforms

Monard Nickl Paternain 2020 Statistical guarantees for Bayesian uncertainty quantification in non-linear inverse problems with Gaussian process priors

Abraham Nickl 20 *On statistical Calderón problems*

Priors for Inverse Problems

Gaussian processes are the favorite priors for linear inverse problems.

Stuart 2010. Inverse problems: A Bayesian perspective

Usually, only the maximum a posteriori (MAP) estimator is kept, so the problem is just minimization with a regularization term.

For some authors, the “bayesian approach” to inverse problems is just minimization of a regularized loss.

Some authors also prove frequentist properties of this MAP estimator.

Monard Nickl Paternain 2017 Efficient Nonparametric Bayesian Inference For X-Ray Transforms

Monard Nickl Paternain 2020 Statistical guarantees for Bayesian uncertainty quantification in non-linear inverse problems with Gaussian process priors

Abraham Nickl 20 *On statistical Calderón problems*

Uninformative Gaussian priors

In abundance of data, most practitioners prefer a noninformative prior, agnostic about the result. A Gaussian process prior is best suited for a linear problem, which encodes basically only information about the regularity of the solutions.
The prior is deduced from the noise model, propagated into the prior by the regularization of the operator.

For the Calderón problem, a natural model is gaussian error in operator space $\mathbb{W}$:

$$Y = \Lambda_h + \epsilon\mathbb{W}$$

Caro Meroño 19 The observational limit of wave packets with noisy measurements

Abraham Nickl 20 On statistical Calderón problems

For a gaussian error in Λ, Abraham Nickl propose a Gaussian prior which a.s. gives values in H^α, $\alpha > 2 + d/2$. The Gaussian process is composed with a link function, since the problem is non linear.

Roininen 14 Whittle-Matérn priors for Bayesian statistical inversion with applications in electrical impedance tomography

Uninformative Gaussian priors

In abundance of data, most practitioners prefer a noninformative prior, agnostic about the result. A Gaussian process prior is best suited for a linear problem, which encodes basically only information about the regularity of the solutions.
The prior is deduced from the noise model, propagated into the prior by the regularization of the operator.
For the Calderón problem, a natural model is gaussian error in operator space $\mathbb{W}$:

$$Y = \Lambda_h + \epsilon\mathbb{W}$$

Caro Meroño 19 The observational limit of wave packets with noisy measurements

Abraham Nickl 20 On statistical Calderón problems

For a gaussian error in Λ, Abraham Nickl propose a Gaussian prior which a.s. gives values in H^α, $\alpha > 2 + d/2$. The Gaussian process is composed with a link function, since the problem is non linear.

Roininen 14 Whittle-Matérn priors for Bayesian statistical inversion with applications in electrical impedance tomography

Informative priors for the Calderón problem

But with such an ill-posed, high-stakes problem as Calderón problem with finitely many sensors, we cannot afford to ignore widely accepted common knowledge data:

Important prior information is available

The mathematical models proposed for Electric Impedance Tomography are the same than those proposed for geophysical imaging.

Furthermore, with so much instability, it is absolutely necessary to quantify the uncertainty.

The Calderón problem needs a proper Bayesian formulation, with a proper informative prior.

Informative priors for the Calderón problem

But with such an ill-posed, high-stakes problem as Calderón problem with finitely many sensors, we cannot afford to ignore widely accepted common knowledge data:

Important prior information is available

The mathematical models proposed for Electric Impedance Tomography are the same than those proposed for geophysical imaging.

Furthermore, with so much instability, it is absolutely necessary to quantify the uncertainty.
The Calderón problem needs a proper Bayesian formulation, with a proper informative prior.

Gaussian Balls around a function?

We would like to consider a **"reference" conductivity** h_0, so that "closer" conductivities are more probable than further away ones. First idea: a "gaussian distribution" in a suitable H^k or RKHS:

$$\min_{\theta} \sum loss\left(f(x_j, \theta), y_j\right) + R(\theta) + \|h_\theta - h_{\theta_0}\|^2$$

If $d(x_1, x_2) > 2\varepsilon$, a standard "$y$-value" norm gives the same distance between $I_{B_\varepsilon(x_1)}$ and $I_{B_\varepsilon(x_2)}$, regardless of the distance between x_1 and x_2.

Gaussian Balls around a function?

We would like to consider a **"reference" conductivity** h_0, so that "closer" conductivities are more probable than further away ones. First idea: a "gaussian distribution" in a suitable H^k or RKHS:

$$\min_{\theta} \sum loss\left(f(x_j, \theta), y_j\right) + R(\theta) + \|h_\theta - h_{\theta_0}\|^2$$

If $d(x_1, x_2) > 2\varepsilon$, a standard "$y$-value" norm gives the same distance between $I_{B_\varepsilon(x_1)}$ and $I_{B_\varepsilon(x_2)}$, regardless of the distance between x_1 and x_2.

Wasserstein distance?

The Wasserstein distance makes sense in medical imaging, because the chest naturally flows, transporting the fluids and tissues. It makes even more sense for geophysical flows.
But Wasserstein distance alone has serious flaws, because the $\mathcal{W}_2$ distance between two densities with different integrals is infinite.
However, it can be useful as a building block with which to compose a more suitable prior.

Inspiration: edit distance of words

Distance(word1, word2) = number of editions required to go from word1 to word2

There are several variants: The *Damerau-Levenshtein* distance, for instance, allows *deletion*, *insertion*, *substitution* and *transposition* of two adjacent characters.

Wasserstein distance?

The Wasserstein distance makes sense in medical imaging, because the chest naturally flows, transporting the fluids and tissues. It makes even more sense for geophysical flows.
But Wasserstein distance alone has serious flaws, because the $\mathcal{W}_2$ distance between two densities with different integrals is infinite.
However, it can be useful as a building block with which to compose a more suitable prior.

Inspiration: edit distance of words

Distance(word1, word2) = number of editions required to go from word1 to word2

There are several variants: The *Damerau-Levenshtein* distance, for instance, allows *deletion*, *insertion*, *substitution* and *transposition* of two adjacent characters.

Edit distance for the Calderón problem

$$\begin{aligned}\min_\theta \quad & \text{loss}\left(D2N(h^\Theta), data\right) \\ & + R(\Theta.A) \\ & + \mathcal{W}_2(Flow^{\Theta.B}\#(h_0^{\Theta.A}), h_0^{\Theta.A})^2 \\ & + \|\tilde{h}^{\Theta.C}\|^2_{RKHS}\end{aligned}$$

Parameters are split into three parts $(\Theta.A, \Theta.B, \Theta.C)$ which play different roles in the conductivity proposal:

$$h^\Theta = Flow(h_0^{\Theta.A}, f^{\Theta.B}) + \tilde{h}^{\Theta.C}$$

- $h_0^{\Theta.A}$ The reference conductivity is parameterized by $\Theta.A \in \mathbb{R}^n$.
- $f^{\Theta.B}$ The flow is parameterized by a neural network.
- $\tilde{h}^{\Theta.C}$ An additive y-space perturbation (FEM, spectral basis, Gaussian process...).

Sample from Wasserstein gaussians?

There are several possible alternatives to quantify the uncertainty in the reconstruction (ABC, continuous normalizing flow, for instance), but we need to be able to sample from the prior.
How to sample from these “Wasserstein balls”?

$$h \to \exp\left(-\mathcal{W}_2(h, h_0)^2\right)$$

First, we must realize that a “density function” from the function space into $\mathbb{R}$ is not a probability measure.

There is no Lebesgue measure in infinite dimensional function spaces

There are many ways of sampling compatible with this “Wasserstein density”, not all equivalent.

Sample from Wasserstein gaussians?

There are several possible alternatives to quantify the uncertainty in the reconstruction (ABC, continuous normalizing flow, for instance), but we need to be able to sample from the prior.
How to sample from these “Wasserstein balls”?

$$h \to \exp\left(-\mathcal{W}_2(h, h_0)^2\right)$$

First, we must realize that a “density function” from the function space into $\mathbb{R}$ is not a probability measure.
There is no Lebesgue measure in infinite dimensional function spaces
There are many ways of sampling compatible with this “Wasserstein density”, not all equivalent.

Sample from Wasserstein gaussians?

There are several possible alternatives to quantify the uncertainty in the reconstruction (ABC, continuous normalizing flow, for instance), but we need to be able to sample from the prior.
How to sample from these “Wasserstein balls”?

$$h \to \exp\left(-\mathcal{W}_2(h, h_0)^2\right)$$

First, we must realize that a “density function” from the function space into $\mathbb{R}$ is not a probability measure.
There is no Lebesgue measure in infinite dimensional function spaces
There are many ways of sampling compatible with this “Wasserstein density”, not all equivalent.

Sampling multivariate normals

Sampling a multivariate normal

1. Identify the distribution for $\|\mathbf{x}\|$, when $\mathbf{x} \sim N(0, I)$.
2. Sample a norm for this distribution $n \sim \chi^2(d)$, where d is the dimension.
3. Choose any method to sample uniformly a point in the sphere $\mathbf{v} \in S^d$.
4. Output $n \cdot \mathbf{v}$

Can do something analogous to this for Wasserstein gaussians?

Problem: this "change to radial coordinates" also requires a measure!

Which is the "number of degrees of freedom"?

Sampling multivariate normals

Sampling a multivariate normal

1. Identify the distribution for $\|\mathbf{x}\|$, when $\mathbf{x} \sim N(0, I)$.
2. Sample a norm for this distribution $n \sim \chi^2(d)$, where d is the dimension.
3. Choose any method to sample uniformly a point in the sphere $\mathbf{v} \in S^d$.
4. Output $n \cdot \mathbf{v}$

Can do something analogous to this for Wasserstein gaussians?
Problem: this “change to radial coordinates” also requires a measure!
Which is the “number of degrees of freedom”?

Sampling Wasserstein gaussians

A proposal for a Wasserstein gaussian of radius r in one dimension:

1. Fix an integer $N > 0$.
2. Extract a positive "length" ℓ from r times a χ^2 with 1 degree of freedom.
3. Split ℓ^2 among the N bins ($\frac{1}{N}\sum b_j^2 = \ell^2$), and randomly choose a sign s_j for each j.
4. Order the resulting numbers $s_j b_j$ in increasing order.
5. Assign those numbers to the quantiles $q_j = Q(\frac{1}{2N} + \frac{j}{N})$ for $j = 1, \ldots, N$.
6. Move each such quantile by the corresponding amount $q'_j = q_j + s_j b_j$.
7. Interpolate linearly between quantiles.

$\Rightarrow$ The total transport cost is $\sqrt{\frac{1}{N}\sum_j b_j^2} = \ell$. But some choices were made :-/

Can we do a similar process in higher dimension, in the lines of yesterday talk by Marc Hallin?

Sampling Wasserstein gaussians

A proposal for a Wasserstein gaussian of radius r in one dimension:

1. Fix an integer $N > 0$.
2. Extract a positive "length" ℓ from r times a χ^2 with 1 degree of freedom.
3. Split ℓ^2 among the N bins ($\frac{1}{N}\sum b_j^2 = \ell^2$), and randomly choose a sign s_j for each j.
4. Order the resulting numbers $s_j b_j$ in increasing order.
5. Assign those numbers to the quantiles $q_j = Q(\frac{1}{2N} + \frac{j}{N})$ for $j = 1, \ldots, N$.
6. Move each such quantile by the corresponding amount $q'_j = q_j + s_j b_j$.
7. Interpolate linearly between quantiles.

$\Rightarrow$ The total transport cost is $\sqrt{\frac{1}{N}\sum_j b_j^2} = \ell$. But some choices were made :-/

Can we do a similar process in higher dimension, in the lines of yesterday talk by Marc Hallin?

Sampling Wasserstein gaussians in 2D

A different proposal for 2D or higher:

1. Fix a kernel and a perturbation magnitude R.
2. Sample $f = (f_1, f_2) \sim RKHS^2$.
3. Flow μ by f, get ν.
4. Compute the optimal transport from μ to ν, and the optimal transport map T.
5. Extract a positive "length" ℓ from R times a χ^2 with 2 degrees of freedom.
6. Instead of following T all the way from μ to ν, follow it just a length ℓ.

Idea behind the scheme:

- The tangent to Wasserstein space are the transport maps.
- The perturbation will cover a neighborhood of μ in a "fair" way.
- So choose a unit tangent vector and follow it $\ell \sim \chi^2$ to get the right statistics.

... and that's about as far as I got

I promised to finish with open questions:

- What is a best probability model for EIT?
- How to sample from Wasserstein Gaussians?
- Should Wasserstein distance play a role in time dependent (dynamic) EIT?

Questions?

Questions?